

XP-002113198

This treatment of the byte stream by TCP is similar to the treatment of a file by the Unix operating system. The Unix kernel does no interpretation whatsoever of the bytes that an application reads or write—that is up to the applications. There is no distinction to the Unix kernel between a binary file or a file containing lines of text.

17.3 TCP Header

P.D. 1994
p. 225-27 = (3)

Recall that TCP data is encapsulated in an IP datagram, as shown in Figure 17.1.

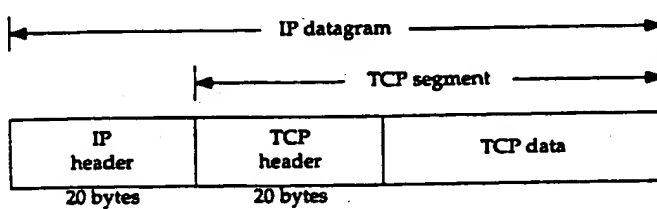


Figure 17.1 Encapsulation of TCP data in an IP datagram.

Figure 17.2 shows the format of the TCP header. Its normal size is 20 bytes, unless options are present.

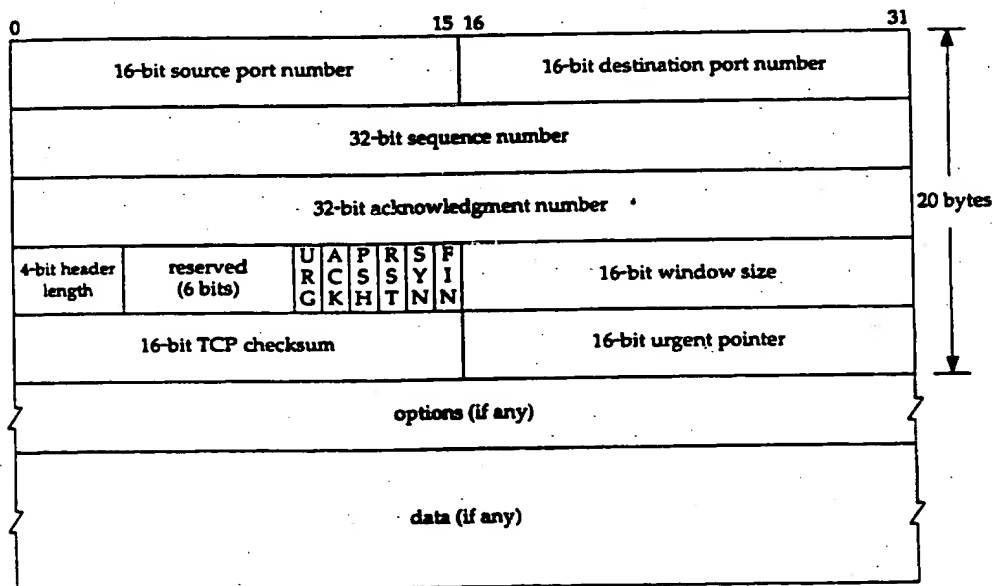


Figure 17.2 TCP header.

Each TCP segment contains the source and destination *port number* to identify the sending and receiving application. These two values, along with the source and destination IP addresses in the IP header, uniquely identify each *connection*.

The combination of an IP address and a port number is sometimes called a *socket*. This term appeared in the original TCP specification (RFC 793), and later it also became used as the name of the Berkeley-derived programming interface (Section 1.15). It is the *socket pair* (the 4-tuple consisting of the client IP address, client port number, server IP address, and server port number) that specifies the two end points that uniquely identify each TCP connection in an internet.

The *sequence number* identifies the byte in the stream of data from the sending TCP to the receiving TCP that the first byte of data in this segment represents. If we consider the stream of bytes flowing in one direction between two applications, TCP numbers each byte with a sequence number. This sequence number is a 32-bit unsigned number that wraps back around to 0 after reaching $2^{32}-1$.

When a new connection is being established, the SYN flag is turned on. The *sequence number field* contains the *initial sequence number* (ISN) chosen by this host for this connection. The sequence number of the first byte of data sent by this host will be the ISN plus one because the SYN flag consumes a sequence number. (We describe additional details on exactly how a connection is established and terminated in the next chapter where we'll see that the FIN flag consumes a sequence number also.)

Since every byte that is exchanged is numbered, the *acknowledgment number* contains the next sequence number that the sender of the acknowledgment expects to receive. This is therefore the sequence number plus 1 of the last successfully received byte of data. This field is valid only if the ACK flag (described below) is on.

Sending an ACK costs nothing because the 32-bit acknowledgment number field is always part of the header, as is the ACK flag. Therefore we'll see that once a connection is established, this field is always set and the ACK flag is always on.

TCP provides a *full-duplex* service to the application layer. This means that data can be flowing in each direction, independent of the other direction. Therefore, each end of a connection must maintain a sequence number of the data flowing in each direction.

TCP can be described as a sliding-window protocol without selective or negative acknowledgments. (The sliding window protocol used for data transmission is described in Section 20.3.) We say that TCP lacks selective acknowledgments because the acknowledgment number in the TCP header means that the sender has successfully received up through but not including that byte. There is currently no way to acknowledge selected pieces of the data stream. For example, if bytes 1–1024 are received OK, and the next segment contains bytes 2049–3072, the receiver cannot acknowledge this new segment. All it can send is an ACK with 1025 as the acknowledgment number. There is no means for negatively acknowledging a segment. For example, if the segment with bytes 1025–2048 did arrive, but had a checksum error, all the receiving TCP can send is an ACK with 1025 as the acknowledgment number. In Section 21.7 we'll see how duplicate acknowledgments can help determine that packets have been lost.

The *header length* gives the length of the header in 32-bit words. This is required because the length of the options field is variable. With a 4-bit field, TCP is limited to a 60-byte header. Without options, however, the normal size is 20 bytes.

There are six flag bits in the TCP header. One or more of them can be turned on at the same time. We briefly mention their use here and discuss each flag in more detail in later chapters.

- URG The *urgent pointer* is valid (Section 20.8).
- ACK The *acknowledgment number* is valid.
- PSH The receiver should pass this data to the application as soon as possible (Section 20.5).
- RST Reset the connection (Section 18.7).
- SYN Synchronize sequence numbers to initiate a connection. This flag and the next are described in Chapter 18.

FIN The sender is finished sending data.

TCP's flow control is provided by each end advertising a *window size*. This is the number of bytes, starting with the one specified by the acknowledgment number field, that the receiver is willing to accept. This is a 16-bit field, limiting the window to 65535 bytes. In Section 24.4 we'll look at the new window scale option that allows this value to be scaled, providing larger windows.

The *checksum* covers the TCP segment: the TCP header and the TCP data. This is a mandatory field that must be calculated and stored by the sender, and then verified by the receiver. The TCP checksum is calculated similar to the UDP checksum, using a pseudo-header as described in Section 11.3.

The *urgent pointer* is valid only if the URG flag is set. This pointer is a positive offset that must be added to the sequence number field of the segment to yield the sequence number of the last byte of urgent data. TCP's urgent mode is a way for the sender to transmit emergency data to the other end. We'll look at this feature in Section 20.8.

The most common *option* field is the maximum segment size option, called the MSS. Each end of a connection normally specifies this option on the first segment exchanged (the one with the SYN flag set to establish the connection). It specifies the maximum sized segment that the sender wants to receive. We describe the MSS option in more detail in Section 18.4, and some of the other TCP options in Chapter 24.

In Figure 17.2 we note that the data portion of the TCP segment is optional. We'll see in Chapter 18 that when a connection is established, and when a connection is terminated, segments are exchanged that contain only the TCP header with possible options. A header without any data is also used to acknowledge received data; if there is no data to be transmitted in that direction. There are also some cases dealing with timeouts when a segment can be sent without any data.

17.4 Summary

TCP provides a reliable, connection-oriented, byte stream, transport layer service. We looked briefly at all the fields in the TCP header and will examine them in detail in the following chapters.

104

WASHINGTON, D.C. 20540

OFFICE OF THE SECRETARY OF DEFENSE

DEPARTMENT OF DEFENSE

ATTENTION: SECRETARY

ATTENTION: SECRETARY

DATE: 10/10/70

TO: SECRETARY OF DEFENSE

FROM: [illegible]

SUBJECT: [illegible]

10/10/70

10/10/70

10/10/70

10/10/70

THIS PAGE BLANK (USPTO)

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70

10/10/70